OBJECT CONTAINER TRANSFER SYSTEM AND METHOD IN AN OBJECT BASED COMPUTER OPERATING SYSTEM

The present invention relates generally to multitasking digital computer systems and particularly to methods and systems for managing the data structures used by a multitasking digital computer system.

BACKGROUND OF THE INVENTION

Large computer systems generally allow many users to simultaneously use a single computer's resources. Such systems are herein called multitasking digitial computer systems. Such computers include virtually all mainframe computers and most minicomputers.

One of the primary jobs of the operating system for a multitasking computer system is to support and keep track of the operations of a multiplicity of users who are running numerous concurrent processes. Thus the computer's operating system must have data structures which represent the status of each user. Such status information includes the memory and other resources being used by each user process.

If every user process were completely independent, had its own dedicated resources, and there were no concerns about which resources each process could use, operating systems could be relatively simple. However in actuality, computer resources are shared and many user processes need to access commonly used or owned resources. In fact, each user may generate a number of execution threads which run simultaneously and which need to be able to share resources and to communicate with other ones of the user's threads.

Another concern in multitasking computer systems is security and data integrity. Ideally, the computer system should provide an access security system which enables each user to control the extent or amount of sharing of information that belongs to the user. Further, the system should provide several types of protection. For example, when multiple processes are allowed access to a resource, the identity of each process which attempts to access the resource should be tested to determine if that particular process is authorized to access the resource. The system of access control should also provide limited "visibility" of computer resources so that an unauthorized user cannot obtain information about another user by repeated attempts to access resources with various names. In addition, to protect data integrity, the system must protect against simultaneous accesses by different authorized processes.

Yet another concern of multitasking operating systems is clearing the system of "objects" (i.e., files and data structures) which are no longer needed by any of the system users. Ideally, the system should also be able to automatically deallocate resources, such as input/output devices, no longer needed by a process.

SUMMARY OF THE INVENTION

In summary, the present invention is an object based operating system for a multitasking computer system. The present invention, which is also called an object based architecture, is "object based" because it provides objects which represent the architecture or interrelationships of the system's resources. The present invention provides an extensible, yet rigorous framework for the definition and manipulation of object data structures.

Objects, generally, are data structures which store information about the user processes running in the system, and which act as gateways to using the system's resources. Resources, generally, include sets of information, physical devices such as a tape drive, and various programs or "operations". Such resources are not available to a user unless the user has explicit permission to use that resource. More specifically, access to certain objects is required in order to use the corresponding resources of the computer system.

All system objects have a consistent data structure, and a consistent method of defining the operations which apply to each type of object. As a result, it is relatively easy to add a new type of system object to the operating system, or to change an existing system object.

Another feature of the present invention is a multifaceted access control system. The object based operating system of the present invention supports multiple levels of visibility, allowing objects to be operated on only by processes with the object's range of visibility. This allows objects to be made private to a process, shared by all processes within a job, or visible to all processes within the system.

In addition to visibility control, access to each object is controlled through an access control list which specifies the processes authorized to access the object, and the types of access that are allowed. An object with a restricted access control list can be associated with a "privileged operation", thereby restricting use of the privileged operation to those user processes authorized to access the corresponding object. An object can furthermore be allocated to a specified job or process to protect the object from use by others, thereby denying access by otherwise authorized processes.

Yet another feature of the present invention concerns "waitable objects", which are objects used to synchronize the operation of one or more processes with one another or with specified events. The present invention provides routines for generating new types of waitable objects without modifying the operating system's kernel. More particularly, a set of several different types of predefined kernel synchronization primitives can be embedded in user defined objects, thereby enabling ordinary programmers and system users to define and generate waitable objects.

BRIEF DESCRIPTION OF THE DRAWINGS

Additional objects and features of the invention will be more readily apparent from the following detailed description and appended claims when taken in conjunction with the drawings, in which:

FIG. 1 is a block diagram of a computer with a multitasking operation system.

FIG. 2 is a block diagram of the virtual memory spaces of several concurrently running user processes.

FIG. 3 is a block diagram showing how data structure objects in the system are organized in a three level hierarchy.

FIG. 4 is a block diagram showing the hierarchical relationship between a user object, a job, the processes for a job, and the execution threads for a process.

FIG. 5 is a block diagram showing the range of objects visible to a particular execution thread.

FIG. 6 is a block diagram of the container directory and object container data structures at one level of the three level hierarchy shown in FIG. 3. FIG. 6A is a